MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

MAC-TR-13

A NEW METHODOLOGY FOR COMPUTER SIMULATION*

by Martin Greenberger

ABSTRACT

Computer simulation is a cooperative venture between researcher and information processor, but the processor's role customarily begins too late. The researcher can benefit substantially by bringing the computer up into the earlier, creative phases of the simulation process. An on-line computer system that makes this possible is described.

*Presented before the Conference on Computer Methods in the Analysis of Large-Scale Social Systems, sponsored by the Joint Center for Urban Studies of the Massachusetts Institute of Technology and Harvard University, October 19-21, 1964.

Work on the OPS-2 system was supported by Project MAC, an M.I.T. research program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Number nonr-4102(01). Reproduction in whole or in part is permitted for any purpose of the United States Government.

CONTENTS

This empty page was substituted for a blank page in the original document.

Introduction

I shall begin by drawing some boundaries around my subject. Simulation, broadly interpreted, is model building. Model building, generously viewed, is theorizing. And theorizing, in its respectable connotation, is at the heart of all science. Thus, the subject of simulation must be circumscribed.

It helps to precede the word simulation by "computer". At the present time, this modifier restricts the domain of reference appreciably, since it implies that the theory is formed to be examined or tested by machine. The modifier may not continue to be so restrictive in the future, however, as the computer's potential becomes better utilized. You might say that this is one of the goals of M.I.T.'s Project MAC,[15] where much of the work underway is devoted to making the computer more generally useful to the researcher.

Researchers in several fields, principally the behavioral sciences, are beginning to find that computer programs make good models of the phenomena they study. The language of the computer is much more versatile than the language of mathematics for many purposes. The implications of a computer model can be inferred directly, simply by executing the model on the machine for which it was programmed. No great amount of mathematical finesse is required.

A computer model, furthermore, can exhibit behavior reminiscent of intelligent human activity, in manner and detail as well as in consequence. That is, information processes programmed on a computer can operate in ways which seem analogous to the thought patterns evidenced by human subjects on their protocols. These facts are starting to open new doors in psychological research.

Not all computer simulations are conceived for the purpose of theoretical research. One of our activities at Project MAC has been the simulation of our time-shared, multi-user computer operation. The point of this simulation is to

gain understanding of the operation and find a rationale for allocating time grants to users efficiently. Monitoring the operation has helped serve the same end, but simulation permits a more varied, controlled, and complete range of experience than does observation. Analytical methods have also shed some light, but permit one to go only so far.

A similar example is simulation of a job shop to guide scheduling decisions. In these kinds of studies, analysis, observation and simulation ideally go hand-in-hand. In the future they may even be blended in with the operation itself, but such thoughts take us ahead of the story.

For our purposes, simulation is a way of using the computer to produce a reasonable likeness of the behavior of a system under study. The likeness is obtained from a scaled-down abstraction of the real system, often in the form of a dynamic model. The model is based on the simulator's concept of what the key elements of the system are, and of how they operate and interact.

In forming his model, the simulator generally strives for maximum parsimony with minimum distortion. But conciseness is not often easy to attain, and most computer models are unavoidably longwinded. This has prompted Herbert Simon to label them "garrulous", in contrast to the really parsimonious models of Newtonian mechanics. The intricate models of organic chemistry are also garrulous by Simon's distinction.[12] The classification is not meant to be derogatory--just a fact of life.

Perhaps the best way to define the general goal of the simulator is maximum fidelity, especially in certain critical dimensions, with minimum complexity. How well this objective is met in practice is one of the items which shall concern us.

Before getting started on the methodological discussion, I had better present some credentials. My union card in simulation goes back 8 years to a study of the consumer sector of the American economy with Guy Orcutt.[9] My union card in computers goes back 14 years to an apprenticeship on the Mark I at Harvard. I have been employed regularly in these fields during all of the intervening time (although I admit to having missed a few years of union dues). This experience does not qualify me as a master craftsman, but it does mean that I have had enough hard knocks to jostle my innocence and jar my idealism a bit. I believe I speak as one whose hopes for computer simulation have been seasoned over the years, but not shattered.

While the tone is still personal, I would like to give an illustration which will help me argue one or two methodological points. It has to do with data collection, and is an example with ~~which~~ I feel particularly at home. I think many of you may share the feeling.

<u>**Heating a House: A Case Study in Data Collection**</u>

Three-and-one-half years ago my wife and I purchased a very large, old, three-story, 16-room, hill-top, uninsulated house. (The last characteristic was not mentioned in the real estate advertisement). It was clear to us from the start that heating the house would be a major expense.* The heating plant consists of two gas furnaces, one blowing hot air to nineteen registers, and the other circulating hot water to ten radiators. Each furnace is controlled by its own clock-thermostat.

---

*As a matter of information, I believe that Boston gas rates are among the highest in the world. In 1961, 300 ccf (hundred cubic feet) of gas cost $49.10 in Boston, $47.67 in New York City, and $21.23 in Pittsburgh.

During our first winter in the house, partly in order to divert my attention from the growing gas bills, and partly in the hope of finding ways to alleviate the cost burden, I began to collect daily figures on gas consumption. The result of this undertaking was forty days of data. The data exhibited considerable variance, fluctuating from a daily high of 38 ccf to a daily low of 17 ccf. A spread this wide gives the optimist some hope of finding measures to keep consumption as low as possible without sacrificing comfort.

Let us view the matter as an eager student of simulation might. What we have here is a complicated system:the house together with the assorted apparatus for producing, distributing and controlling heat. The manipulable or instrument variables include storm windows, thermostats, an aquastat, radiator valves, air registers, and dampers. There are also nine fireplaces, but assume for simplicity that they are all closed off. The target variables are the temperatures in each of the rooms, maintenance and service costs, and the monthly gas bills. Altering the number, location, or setting of any of the instrument variables is a means for adjusting the target variables.

To simulate this system, we must decide which variables deserve inclusion, and we must determine relationships which link the variables. An obvious variable to include is outdoor temperature, since we know it directly affects the heat loss radiated from the house. In the parlance of the model-building trade, outdoor temperature is an exogenous variable of decided importance.

Figure 1 displays the forty daily gas consumptions, each one plotted against the mean outdoor temperature on the corresponding day. As would be expected, the gas consumption depends inversely on mean temperature. The lower the temperature, the more gas consumed, and conversely.

On first try we might fit a straight regression line to the points of Figure 1, as shown. This line provides an initial relationship for the model. By the way, it actually is not a bad fit as regression lines go. It yields a numerical correlation coefficient of just a shade under .9. But with more information, we can do better.

My wife and I both go to work on weekdays, and during that first winter we did not have children at home to keep warm. I, therefore, had the habit of turning the settings of the two thermostats down 10° when we left in the morning. The settings automatically reverted to normal for our return in the evening. Let us call this policy A. On the weekends, when we were at home, and on Wednesdays, when a lady came to clean, I kept the thermostats at the same settings throughout the day. Let us call this policy B. In both policies, I lowered the settings overnight.

If we now separate the points of Figure 1 into those associated with policy A (Figure 2) and those associated with policy B (Figure 3), and neglect the remaining points, we obtain surprisingly close fits to each of two smooth curves. In Figure 2, only the three hollow points are substantially off the curve. All three of these points lie above the curve, and all three of them correspond to Tuesdays.

The points of Figure 3 show a little more variation. The three hollow points falling beneath the curve all correspond to Wednesdays, and two of these three Wednesdays happen to be the day after two of the three Tuesdays cited above. Since the latter have larger consumptions than their curve would predict, while the former have smaller consumptions, my guess is that I made incorrect meter readings on the corresponding two Tuesday nights.

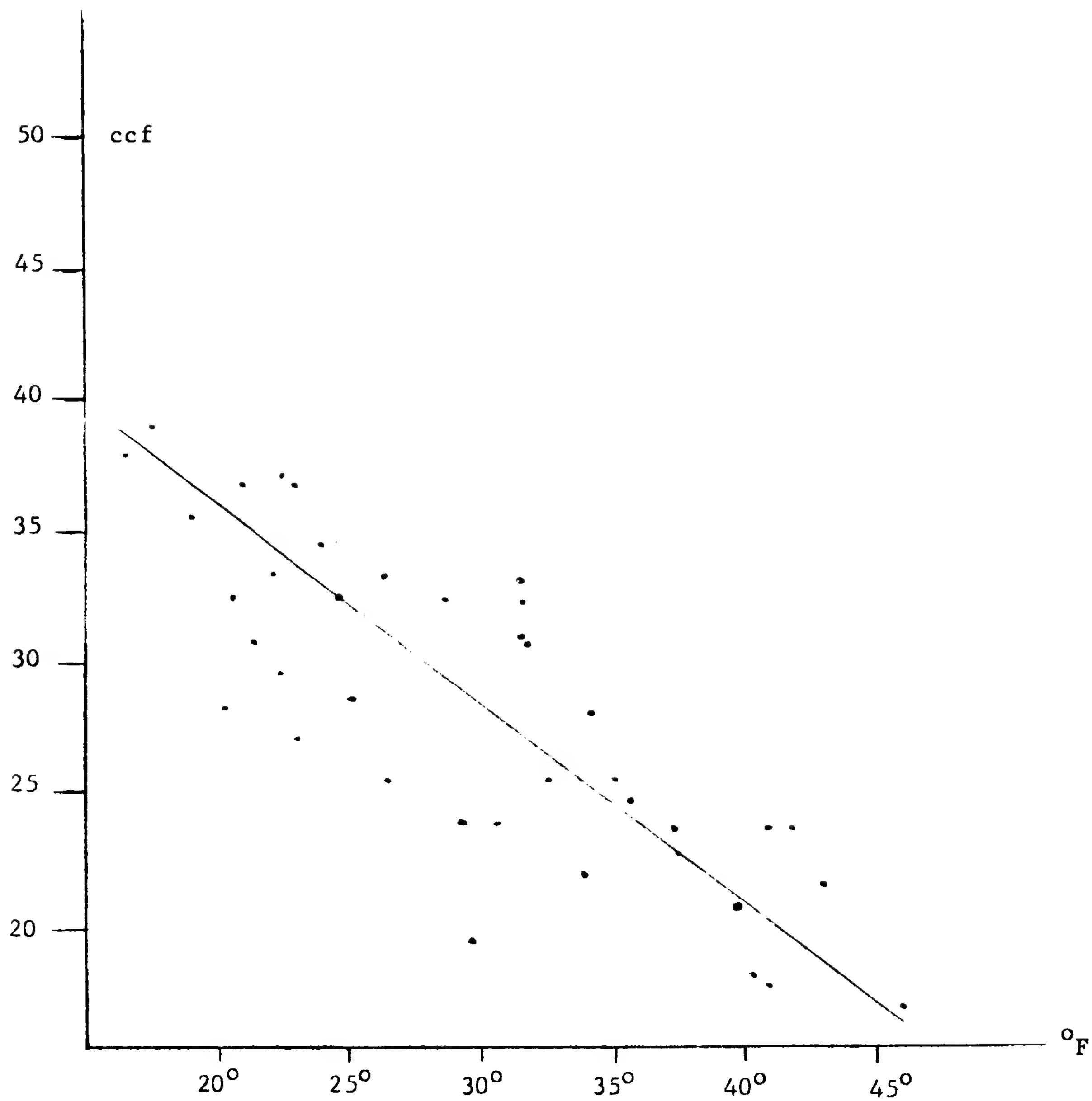


Figure 1: Daily gas consumption 40 days during December 1961 and January 1962. (Gas consumption measured in 100 cubic feet, temperature in degrees Fahrenheit).

If we superimpose the curves of Figures 2 and 3 on each other, as in Figure 4, we notice that they have opposite concavities, they bend toward each other at the ends (20° and 45°), and they depart from each other at the middle (30° to 35°). This is not the appropriate place to speculate on the physical reasons for this behavior, but we can note the economic implications, as given in Figure 5. The greatest potential saving obtained from using policy A rather than policy B occurs in the middle range of temperature, and this saving decreases steadily as either of the extreme temperature ranges is approached. At temperatures below 15° and above 50°, we might conclude that both policies cost about the same.

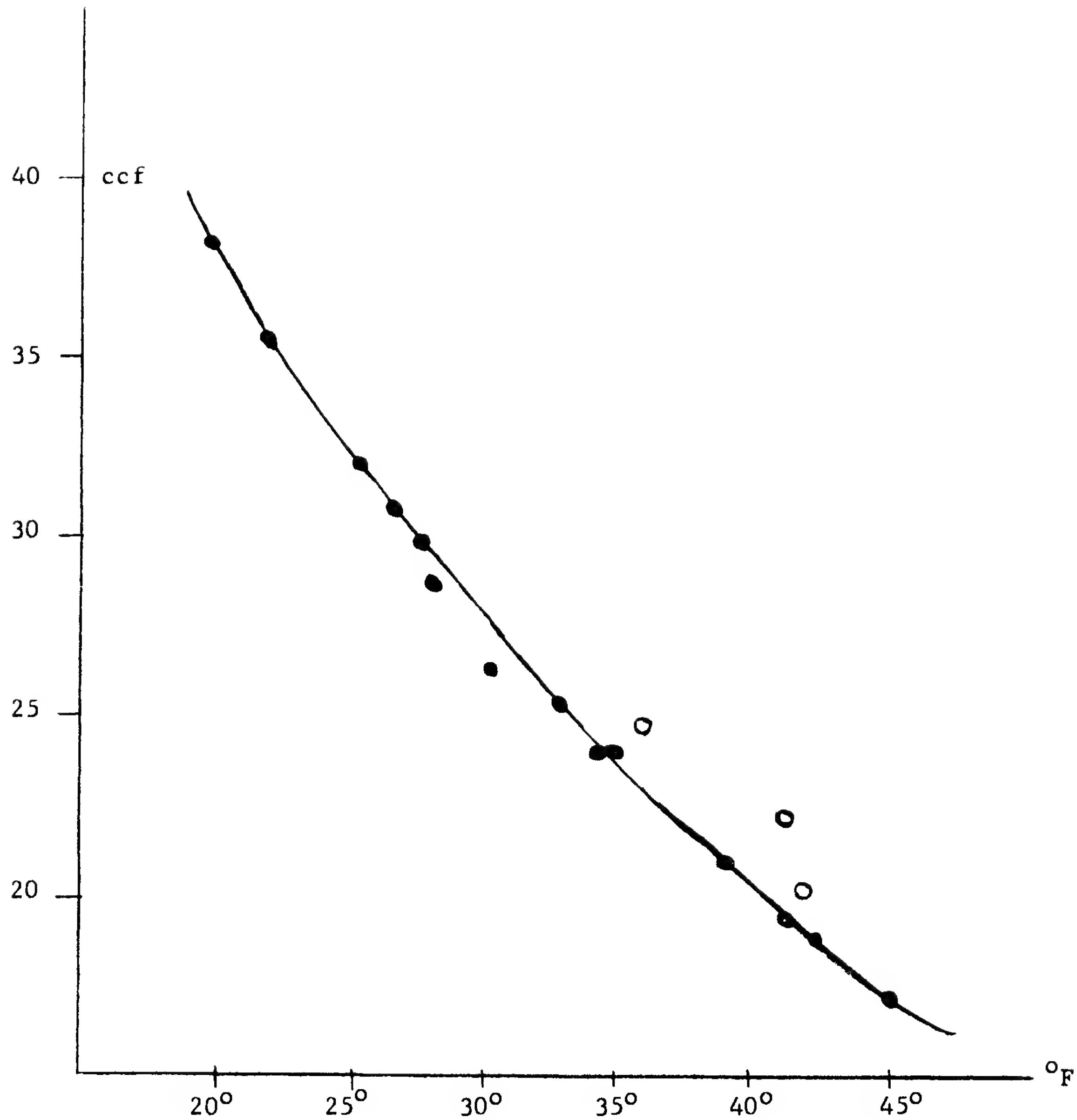


Figure 2: Days from Figure 1 on which policy A was used.

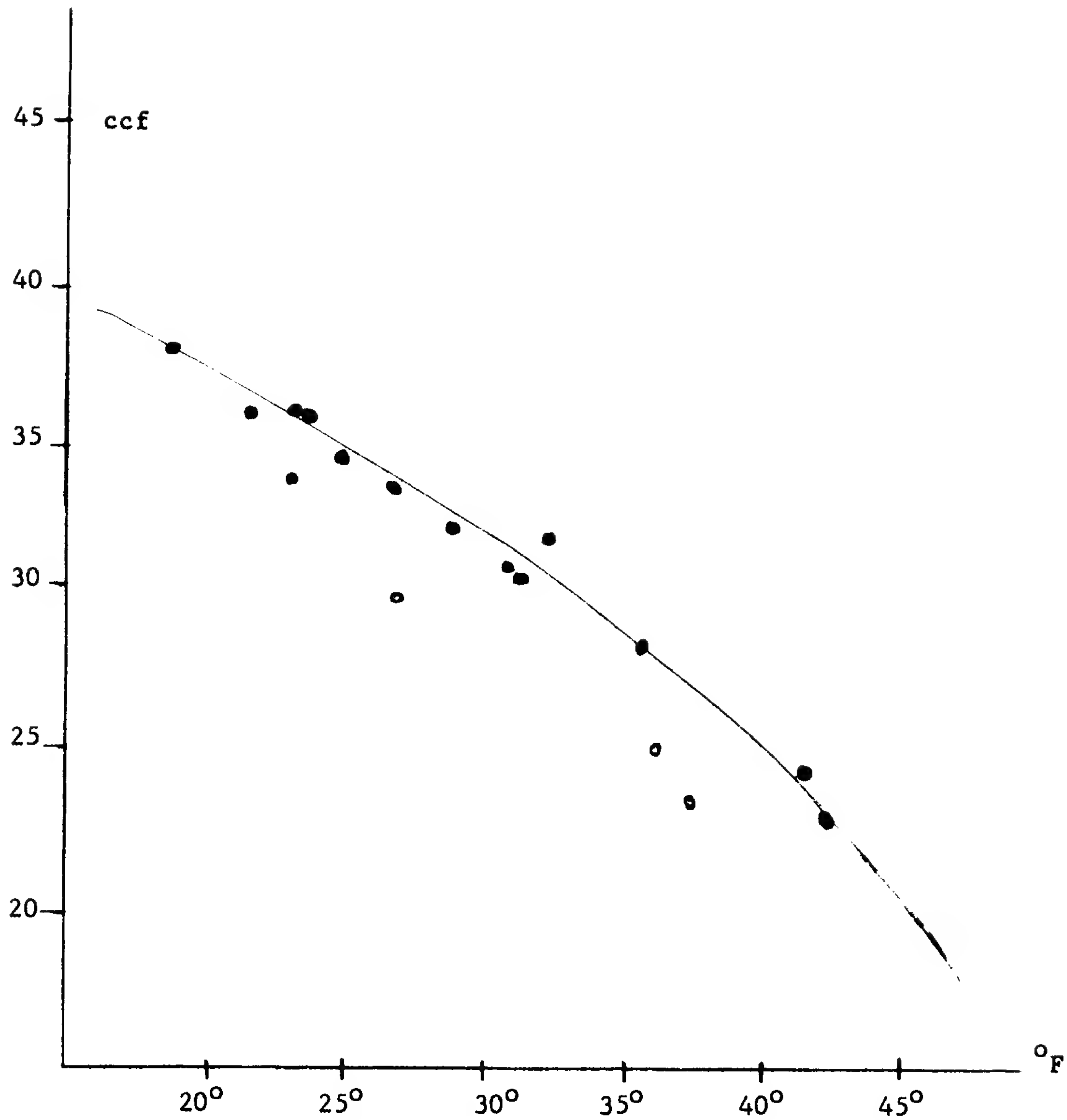
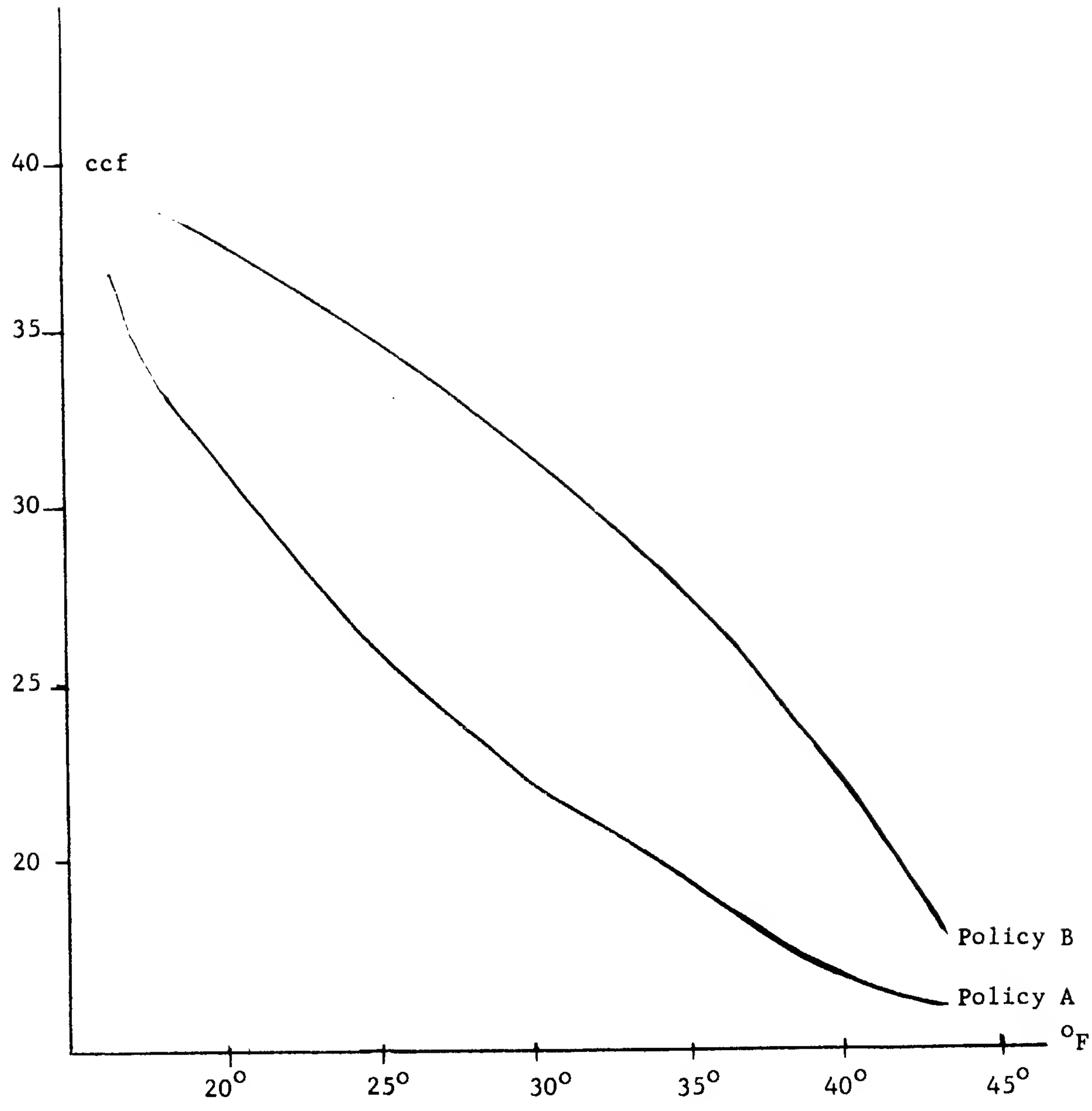


Figure 3: Days from Figure 1 on which policy B was used.



<u>Figure 4</u>: Superposition of Figures 2 and 3.

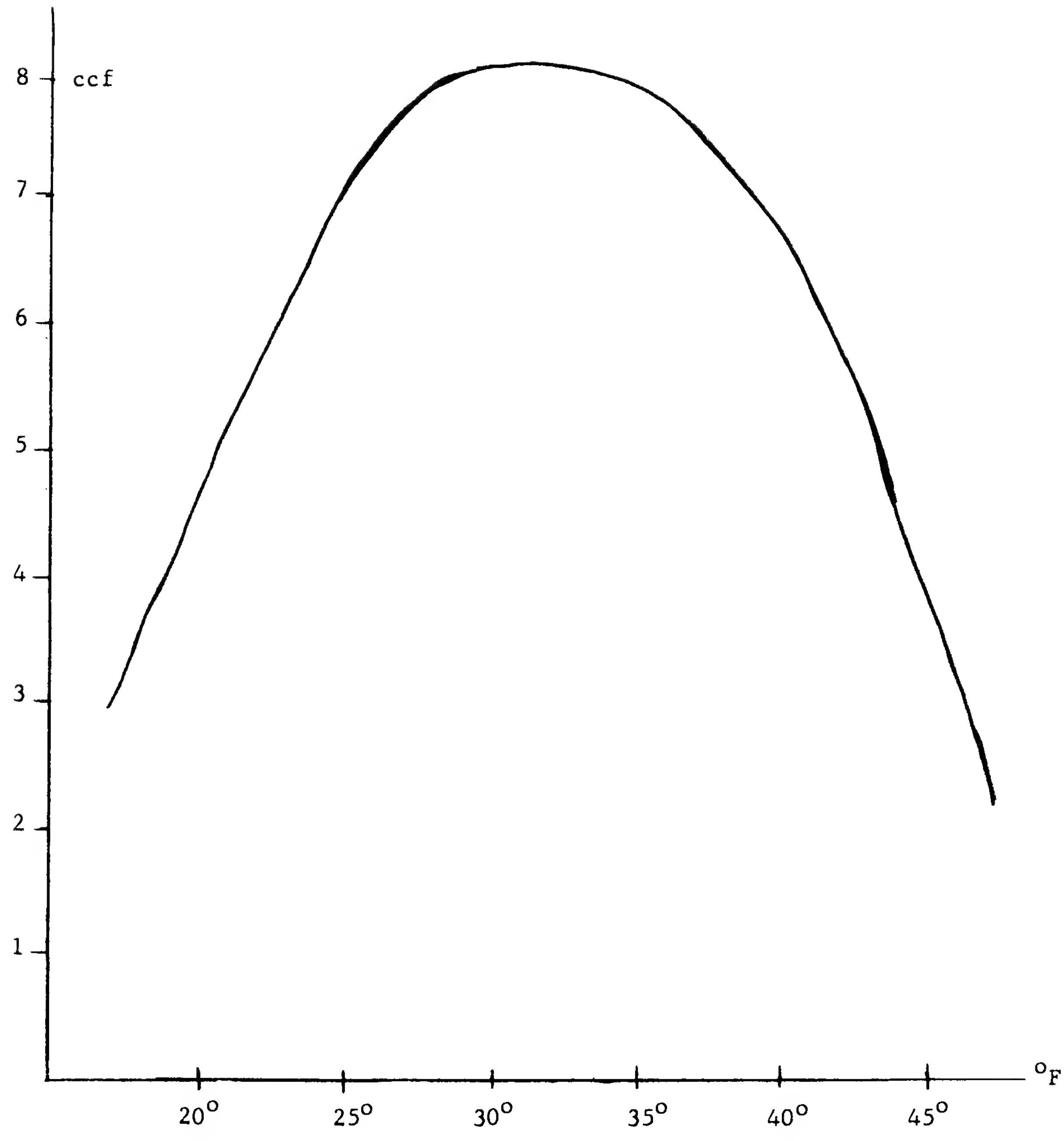


Figure 5: Absolute difference between consumptions resulting from policies A and B.

Some Methodological Issues

The finding of opposite concavities was interesting from an academic point of view, and it even turned out to have some practical utility. Without over-doing its importance, and without stretching the analogy with a simulation study too far, I believe we can extract a few simple lessons from the example.

In the first place, a careful analysis of available data can assist us in choosing variables and relationships for a simulation model that we are building. The analysis not only serves for guidance, but also helps keep us honest. If we had gone on to simulate the heating system in detail, we would not have rested until the model exhibited the property of opposite concavities. Conversely, if the results of the simulation showed certain other peculiarities, we would look for corresponding features in the data, and we would alter our model if they were not to be found.

This is an obvious point, and most simulators accept it implicitly. What it means is that the simulator benefits from frequent and easy travel between model and data throughout model development. But this is easier said than accomplished. Too often data analysis and model development occur in separate, prolonged efforts, one after the other. Only when the model is complete, if then, does the simulator have the interest or resources to return to the data for verification. By then it may be too late. The final complexity of the model may make serious validation impractical.

The point is worth pursuing a bit further. Simulations are often called dynamic, and what this usually means is that time is one of the variables of the model. We might consider a second use of dynamic to describe the formation of the model, rather than its execution. The second kind of dynamic simulation

is constructed over a period of time in an incremental way, with new increments and old increments forming and evolving from successive reconfrontations of the partial model with the data. This approach can produce a continuous stream of fresh insights and can take full advantage of new data as it becomes available--a meaningful feature, especially in a forecasting type of model.

The incremental approach facilitates checking the model, as we have noted, and it allows the simulator to build his understanding of the model in comprehensible segments, as he builds the model itself. It guards against unnecessary arbitrariness, and pinpoints deficiencies both in the model and in the data. The awareness of information deficiencies, modified by a knowledge of which parts of the model are most critical or sensitive, can provide valuable guidance to data collection efforts.

The picture that suggests itself is a research loop composed of several phases: data collection, data analysis, model formulation, programming, testing, adjusting, and running. The phases are not placed serially in time, one starting after the preceding one has been completed, but are rather continuously traversed in a gradual convergence to the final simulation. Actually, a "final simulation" need no longer be the primary motive since research dividends now are being paid all along the way.

Ideally, the simulation is never more complicated than necessary for the ends it is to serve. Excessive complexity is not the bugbear that it can be when the model is designed in one continuous, determined, somewhat unquestioning effort. Overfitting to the data is still a possible pitfall, but the danger signs are now more readily distinguished.

The temptations toward overcomplexity in a computer simulation are very real and very understandable. They stem from the versatility of the computer's language which I referred to earlier. It is so simple to be literal about the way things happen in the real system when modeling the system on a computer. Stochastic effects are easy to copy, nonlinearities offer no particular problem, interaction is not a challenge, and behavior can be portrayed to any level of detail, however fine. Superficially, the only reason to exercise any restraint at all is to keep the formulation within the space and time capabilities of the machine.

On a more thoughtful plane, it is clear that complexity which cannot be unraveled obstructs understanding and validation. This is already a problem in the application of simulation to scientific research, and will become a problem in a practical context, as well, when the results of simulations begin to be used routinely to support business, military, and legislative decisions. How do you mediate between two conflicting sets of results when you are unable to anatomize the models that produced them?

This brings us to another obviously important, but frequently neglected point. Behavioral simulations should be conducted by behavioral scientists, of course. And yet we see programmers, mathematicians, and engineers performing much more than a service function in behavioral simulations.

Again the reasons are clear. Getting a simulation built and running in the conventional way requires analytical and programming talents which are too specialized and technical for many behavioral scientists. Yet, only the behavioral scientist has the first hand knowledge of the data and the intimate familiarity with the phenomena vital to a successful simulation. Subtleties in the data can make the difference between victory and defeat, depending upon whether they are recognized or overlooked.

What is victory? That clearly depends on one's set of values. I am thinking of the values of the behavioral scientist, which are chiefly keyed to deeper understanding of the phenomena being studied. Programmers and mathematicians have a different set of values. They characteristically relish questions of structure, style, and technique, and they celebrate when the simulation first runs. Of course, behavioral scientists are not immune from some excitement at such moments, too, but ordinarily their work is then just beginning.

By the way, who else, including the most clever statistician or cryptographer, could have resolved the data of Figure 1 into Figures 2 and 3? I simply happened to be the only one who knew the data well enough to recognize that the circumstances of collection were dichotomous. It is true that I could have imparted this knowledge to a heating consultant, but I may not have been thinking on that level if I were not conducting the study, myself. Also a heating consultant could not be expected to be as cost-minded in making experiments as I was, since he would not be as personally identified with the problem.

A Computer System for Simulation Research

This is all well and good, and may even serve to convince a few of the people who need convincing that the present way of simulating leaves something to be desired. But where do we go from here? Where do we look for improvements?

These questions have been on my mind for a number of years. It seems to me that anyone who has been deeply involved in a large-scale simulation study must have had similar concerns at some point. What is needed is a new kind of relationship between the researcher and his computer; a relationship characterized by a high degree of accessibility, close coupling, and fast interaction; a relationship that treats the computer as more than a workhorse for running simulations; a relationship that brings the computer up into the earlier, more creative phases of the research process.

This kind of relationship starts to become plausible at places like Project MAC, where a large computer system is being utilized simultaneously by many users at terminals distributed over wide geographical areas. Transfer of information to and from the computer is via conventional telephone lines. Each user is able to be in continual communication with his program and his data without suffering the high cost and tension associated with monopolizing an expensive machine.[15]

Multi-access computers are a first, and very important step. The second major requirement is the development of an on-line programming system that enables the researcher to modify his program as he operates it, and switch flexibly between study of his data and construction of his model. This allows him to build up programming packages as he needs them and as his comprehension of his problem grows. The computer helps to guide him along alternate paths of inquiry, and the researcher moves back and forth smoothly between machine-aided analysis of his problem and gradual synthesis of a solution.

My group at Project MAC has been working for the past year to develop an on-line system with these features. We call it the OPS system, and its current implementation is labeled OPS-2. Because of the generality and simplicity of the OPS concept, the system has actually been applied to a wide variety of activities besides simulation research. Our experience in the simulation area up to now has been very encouraging, but is still only preliminary.

In describing the OPS system, I shall repeat a few of the points mentioned earlier, not so much for emphasis, but simply to make the description relatively self-contained. The description will have to be brief. As with any programming system, real understanding can only be obtained through use, and we hope to have a self-teaching manual available for that purpose in the near future.

OPS-2

OPS-2 is a new research tool with broad scope and flexibility. It links the user with the computer in a laboratory environment that makes mutual interaction both simple and powerful.

OPS-2 is an on-line system. It is based upon our newly acquired ability to make large computers personally accessible to a community of many simultaneous users.

There is a two-part premise implicit in the rationale of the on-line system. First, the computer often can and should take an important part in man's creative process during the origination of his ideas and the formulation of his model. Second, and conversely, man often can and should play a key role in the computer's process to guide the execution and fulfillment of his designs.

Clearly, this premise is not equally valid for every human creative process. But it does hold for a surprisingly wide class of processes.

One illustration is the development of a computer simulation. In the early stages, when the researcher is deciding upon the form and content of his model, the computer can assist in data analysis and statistical regressions. It also can help decide among alternate formulations of subparts of the model by deducing their implications vis-à-vis the real-life data, suitably transformed when required.

The computer is at the researcher's side, in effect, like his manuals, journals, notebook, and telephone. If the researcher wonders whether log t (the logarithm of time) is a more revealing variable than t, an answer may be forthcoming from the computer within minutes.

The simulation model takes shape right from the start of the process. As more and more parts are added, the researcher runs them in combination, as well as singly, and his understanding of the growing model also grows. There is no longer a sharp dividing line between the phases of data analysis, formulation, programming, running, and validation. All begin together and continue intertwined throughout the process.

In the later stages of the simulation, when the dominant activity is making runs, there are occasional returns to data analysis, formulation, reformulation, programming, reprogramming, and validation. The programming is far easier to accomplish than programming has been traditionally, and the researcher finds it convenient to do most of it himself. Programming is no longer strictly detached from the rest of the creative process.

The researcher maintains his active role through to completion. He may even choose to include himself (or others) as live elements of the simulation in order to feed it with semi-realistic behavior. This latter device has been practiced for many years, but seldom has it fitted into a system so naturally as it does with multi-terminal, personally accessible computer systems.

Data analysis, like programming and other forms of problem solving, can be a creative process in its own right, inside or outside of the context of simulation. And simulation can be part of a larger process. A man-machine system for scheduling a job shop, a real-time operation for controlling the traffic of a metropolis, an automated security or commodity exchange, and a computer-administered credit center on the regional level, are all processes which can, and probably should have simulation elements as basic components.

The OPS system provides a basis for building up such processes. It is open-ended and modular in a very fundamental sense. The user can add his own parts over a period of days or months as he increases his understanding of his problem.

The OPS system is relatively free of rules and formats. The user creates his own language and his own conventions. He has the widest latitude to express his problem in its natural terms and to be inventive. Gradually his system takes on an individual character appropriate to the purpose it is to serve.

As a result, OPS-2 covers a broad spectrum of possible applications, including all of the processes previously discussed. For hybrid systems that combine two or more of these processes, OPS-2 provides a simple, integrated framework.

This fact can have benefits in economy as well as in research effectiveness. It can speed up development effort and avoid duplication. Simulation elements that are constructed to test out the prototype of a real-time system can evolve into the actual operating elements of the system. They can also continue to serve as simulation elements, to provide the system with a means for monitoring and extrapolating its own performance during operation.

The basic structure of the OPS system is easy to visualize. There is a body of data located in common storage, and there is a set of operators which operate on this data. The data consists of lists, multi-dimensional arrays, and single elements. Reference to the data is symbolic, and an index of symbols and dimensions is incorporated as part of common storage.

Reference to the operators is also symbolic. There is a central mechanism for executing operators and compounding them in flexible combinations.

The user can create his own symbols and his own mapping of common storage by means of standard operators. He can also create his own operators and add them without limit to the set of standard operators supplied to him.

Operators are functional subroutines programmed in any language that the computer can compile, such as FORTRAN, MAD, or FAP. Each operator can have a number of modifiable parameters associated with it, and thereby may be capable of a range of functions.

Consider the vector operator called TYPE, for example. TYPE is one of a number of operators which form a vector-processing package that we have been using a great deal. Its first parameter is a word such as INTO, OUT, or OVER. This distinguishes whether the vector or vectors are being entered, displayed, or edited, respectively, or any of several other possibilities.

The second parameter of TYPE is the name of the vector, and the third parameter is the name of a second vector, if more than one is being referenced. The fourth parameter is the number of an element, if the operator is to begin with a specific component of the vector (s). And so on. Only as many parameters as pertain need be specified.

Other operators in the vector package include one that does polynomial fits and multiple linear regressions, one that performs a wide class of vector transformations, one that transfers vector data to and from secondary storage, and one that plots functions on a cathode-ray tube. Each of these operators has three or more parameters associated with it. More powerful operators for general array processing are also available as a standard package.

As another example, in an on-line [illegible] the operators are event [illegible] for [illegible] operators [illegible] line to any line, any number of [illegible] as [illegible] simultaneous equations, one [illegible] by the settings of an internal [illegible] or creating [illegible] for each operator in the [illegible] of operators and their parameters [illegible] to alter any parameter of any operator and to [illegible] an entire line of the N-OP. The [illegible] by an M.

Most of the direct commands to the system are carried out by the system itself, without reference to any operator. By enclosing a set of such commands

TYPE OUT 4

for example, types out all of the components of vector 4. The user can place this operator in a K-OP, with or without execution. He can execute a K-OP from any line to any line, any number of times, with or without entry to the system between execution of successive operators. He can have results of the execution displayed as he goes along, or only at specified lines, or he can suppress results altogether. And similar options are available to him for the display of both guidelines of an operator and the parameters of a K-OP.

Which of these actions is performed, and how it is performed, is determined by the settings of an internal bank of programmed switches. The user can alter the setting of any of these switches. Typing G turns on guidelines; NR turns off results; P turns on parameters; C causes operators to be entered into a K-OP; NX suppresses execution; and so on.

The user can also request a status reading by typing STATUS. This informs him of the switch and line settings currently in effect. He may change the line number whenever he chooses. Typing

L 120

for example moves the line designator to line 120.

Switch settings may be compounded even more easily than operators. All the user need do is type MODE and a number, whenever he wishes to preserve the group of settings in effect at a particular time. Thereafter he can reestablish this group of settings, at any point, simply by specifying the proper number, preceded by an M.

Most of the direct commands to the system are carried out by the system itself, without reference to any operator. By enclosing a set of such commands

in parentheses as he types them, the user indicates to the system that they are to be <u>deferred</u>, not carried out immediately as is customary. A set of deferred commands may be placed on any line of a K-OP, just as though it were a bona fide operator with parameters. During execution of that K-OP, the system will treat the deferred commands, when it reaches them, as though they were being entered by the user from the console. Thus, the user has inserted himself into the K-OP implicitly. He may choose to do this, for example, when a portion of his role in the man-machine process has become sufficiently routine for him to want the computer to assume it.

Deferred commands give a K-OP the ability to skip lines and loop around during execution, and also to modify the parameters and specification of its operators. Certain standard operators give the K-OP all of the other common properties of computer instructions, including the conditional branch. Thus, the K-OP may be thought of as a program whose instructions can

1. perform arbitrarily complex functions
2. modify each other's arguments
3. provide conditional control of program flow
4. introduce and dimension new symbolic variables
5. save and restore entire programs

In particular, one standard operator can put a K-OP away in secondary storage for later reference, or it can bring in a previously saved K-OP to supplement or replace the present one. Like all of the operators, this one can be executed either by the user from his console, or during the execution of a K-OP. In the latter case, the operator is executed without interrupting the run.

Concluding Remarks

In painting the future of simulation as much rosier than its past, I know that I have oversimplified and exaggerated a bit. It is hard not to, when trying to make a point. In particular, I have referred only indirectly to some of the profound intellectual problems that must be resolved if simulation on a computer is to mature into a respected vehicle for scientific research.

In building a simulation, there are typically a number of different approaches that can lead to a working model. An economy, for example, can be modeled in terms of relationships among macroeconomic variables, such as income, saving, and spending; or it can be modeled in terms of the properties of microeconomic decision-making units, such as households, firms, and industries. Relationships can take the shape of difference equations, input-output matrices, or statistical correspondences, with elements that may or may not be probabilistic. These different options are neither mutually exclusive nor totally exhaustive.

The underlying concept of a simulation can sometimes assume either of two almost inverse forms. Like a diffusion process, for instance, a model may have stochastic flow in a deterministic medium; or, like a percolation process, it may have deterministic flow in a random medium. The viewing behavior of a television audience may be phrased as a set of individual viewers, with different viewing characteristics, responding to a programmed environment; or it may be expressed as an interaction of variously described programs competing for their share of a potential audience. The form selected will depend partly on the data available, and not insignificantly on the background and orientation of the researcher.

A simulation may be built from the outside in, as in the erection of a modern skyscraper, or element-by-element in hierarchical combination, as in the formation of social, economic, and political organizations. Some such means of factoring the simulation is important, if not essential, for reasons I have elaborated and for implementation of the incremental approach to modeling that I have proposed.

These are a few of the theoretical issues that are coming into sharper focus as the computer begins to enter the research process in a more intimate way. They would make excellent topical material for a book on model building, but I do not think we shall see a definitive treatment of the subject for some time to come. We are beginning to make strides forward, however, and the strides are certain to grow stronger as we learn to work more cooperatively and insightfully with our information-processing aides. The computer is not going to supplant research activity, but it is going to make a big difference in the form this activity takes.

Acknowledgments

Anthony Gorry, Malcolm Jones, David Ness, Mayer Wantman, and Stephen Whitelaw have all contributed actively to the development of the OPS-2 system.

The system has been programmed on M.I.T.'s time-sharing facilities.[2] The ease with which this was accomplished has been an impressive demonstration of the effectiveness of these facilities. Although OPS-2 at present runs under time-sharing, its concepts apply to any large-memory computer system that emphasizes personal accessibility and man-machine interaction. The concepts become especially attractive in the context of future information utilities.[4]

OPS-2 shares some similar objectives and capabilities with several other on-line systems, and we are indebted to their authors for ideas and inspiration.[1-3,7,10,14] Most of these systems have been developed for a specific class of use, whether it be engineering design, program supervision, mathematical problem solving, or array processing. OPS-2, by contrast, evolves its character as it is applied, and it can remold itself during execution.

The initial version of the OPS system, referred to as OPS-1, was programmed during the Spring of 1963 in an experimental project of an M.I.T. seminar.[6] Its applications have covered a broad spectrum, including: an automated stock exchange, a mechanized system for accounting and budgeting, an array processor, a program supervisor, a project scheduler, an on-line simulation system, and a live FORTRAN programming facility.

OPS-2 is a completely reworked and improved version of the original system. The automated stock exchange, the array processor, and the accounting system are all operational under OPS-2.

References

[1] Biggs, J.M. and Logcher, R.D., Stress: A Problem-Oriented Language for Structural Engineering, Project MAC TR-6, M.I.T., May, 1964.

[2] Corbato, F.J., et al., The Compatible Time-Sharing System: A Programmer's Guide, M.I.T. Press, 1963.

[3] Culler, G.J. and Fried, B.D., An On-Line Computing Center for Scientific Problems, M19-3U3, Thompson Ramo Wooldridge, June, 1963.

[4] Greenberger, M., The Computers of Tomorrow, The Atlantic Monthly, May, 1964.

[5] Greenberger, M. (ed.), Computers and the World of the Future, M.I.T. Press, 1962.

[6] Greenberger, M. et al., The OPS-1 Manual, Project MAC TR-8, May, 1964.

[7] Hellerman, H., Experimental Personalized Array Translator System, Communications of the ACM, Vol. 7, No. 7, July, 1964.

[8] Licklider, J.C.R., and Clark, W.E., On-Line, Man-Computer Communications, AFIPS Proceedings, 1962.

[9] Orcutt, G.H., Greenberger, M., Korbel, J., and Rivlin, A.M., Microanalysis of Socioeconomic Systems--A Simulation Study, Harper, 1961.

[10] Ross, D.T. and Feldman, C.G., Verbal and Graphical Language for the AED System: A Progress Report, Project MAC TR-4, M.I.T., May, 1964.

[11] Simon, H.A., The New Science of Management Decision, Harper, 1960.

[12] Simon, H.A., and Newell, A., Information Processing in Computer and Man, American Scientist, September, 1964.

[13] Sprague, R.E., Electronic Business Systems, Ronald Press, 1962.

[14] Weizenbaum, J., OPL-I: An Open Ended Programming System with CTSS, Project MAC TR-7, April, 1964.

[15] Descriptions and reports of Project MAC are available from its administrative office, M.I.T., 545 Technology Square, Cambridge 39, Massachusetts.